Communicated by Todd Leen

How Dependencies between Successive Examples Affect On-Line Learning

Wim Wiegerinck
Tom Heskes
RWCP Novel Functions SNN† Laboratory,*
Department of Medical Physics and Biophysics, University of Nijmegen,
Geert Grooteplein 21, 6525 EZ Nijmegen, The Netherlands

We study the dynamics of on-line learning for a large class of neural networks and learning rules, including backpropagation for multilayer perceptrons. In this paper, we focus on the case where successive examples are dependent, and we analyze how these dependencies affect the learning process. We define the representation error and the prediction error. The representation error measures how well the environment is represented by the network after learning. The prediction error is the average error that a continually learning network makes on the next example. In the neighborhood of a local minimum of the error surface, we calculate these errors. We find that the more predictable the example presentation, the higher the representation error, i.e, the less accurate the asymptotic representation of the whole environment. Furthermore we study the learning process in the presence of a plateau. Plateaus are flat spots on the error surface, which can severely slow down the learning process. In particular, they are notorious in applications with multilayer perceptrons. Our results, which are confirmed by simulations of a multilayer perceptron learning a chaotic time series using backpropagation, explain how dependencies between examples can help the learning process to escape from a plateau.

1 Introduction

The ability to learn from examples is an essential feature in many neural network applications (Hertz *et al.* 1991; Haykin 1994). Learning from examples enables the network to adapt its parameters or weights to its environment without the need for explicit knowledge of that environment. This paper focuses on a popular learning procedure called on-line learning. In this learning procedure examples from the environment are continually presented to the network at distinct time steps. At each time

*RWCP: Real World Computing Partnership.
†SNN: Dutch Foundation for Neural Networks.

step a small adjustment of the network's weights is made on the basis of the currently presented example. This procedure is iterated as long as the network learns. The idea is that on a larger time scale the small adjustments sum up to a continuous adaptation of the network to the whole environment.

In many applications the network has to be trained with a training set consisting of a finite number of examples. In these applications a strategy is often used where at each step a randomly selected example from the training set is presented. In particular, with large training sets and complex environments successful results have been obtained with this strategy (Brunak *et al.* 1990; Barnard 1992). Characteristic of this kind of learning is that successive examples are independent, i.e., that the probability to select an example at a certain time step is independent of its predecessors. Of course, successive examples in on-line learning do not need to be independent. For example, one can think of an application where the examples are obtained by on-line measurements of an environment. If these examples are directly fed into the neural network, it is likely that successive examples are correlated with each other.

A related example is the use of neural networks for time-series prediction (Lapedes and Farber 1988; Weigend *et al.* 1990; Wong 1991; Weigend and Gershenfeld 1993; Hoptroff 1993). Essentially, the task of these networks is, given the last k data points of the time series, to predict the next data point of the time series. Each example consists of a data point and its k predecessors. There are two obvious ways to train a network "on-line" with these examples. In what we call "randomized learning," successively presented examples are drawn from the time series on arbitrary, randomly chosen times. This makes successively presented examples independent. In the other type of learning, which we call "natural learning," the examples are presented in their natural order, keeping their natural dependencies. In Mpitsos and Burton (1992) and Hondou and Sawada (1994), both types of example presentation are compared for the learning of a one-dimensional chaotic mapping. In their simulations natural learning performs significantly better than randomized learning. This phenomenon, and, more generally, how the presentation order of examples affects the process of on-line learning are the subject of this paper. Understanding these issues is not only interesting from a theoretical point of view, but it may also help to devise better learning strategies.

In this paper we study the dynamics of on-line learning with dependent examples from a general point of view. In Section 2, we define the class of learning rules and the types of stochastic, yet dependent, example presentation which are analyzed in the rest of the paper. Because of the stochasticity in the presentation of examples, on-line learning is a stochastic process. However, since the weight changes at each time step are assumed to be small—in this paper the weight changes scale linearly with a small constant η, the so-called learning parameter—it is possible to give approximate deterministic descriptions of the learning process on

a larger time scale. In lowest order, the learning process can be described by an ordinary differential equation (ODE). The fluctuations, i.e., the differences between the stochastic trajectory of the weights and the ODE, are of order $\sqrt{\eta}$. These fluctuations are described by a covariance matrix. Besides an heuristic (re)derivation of the ODE and the equation for the fluctuations [a more rigorous derivation can be found in Benviste *et al.* (1987) and Kuan and White (1994)], Section 3 also derives in the same heuristic framework an equation for a systematic bias. This bias, which is of order η, describes the lowest order difference between the mean value of the weights and their ODE trajectory. One could interpret the bias as a first order correction to the ODE. With these equations, we will study the effect of dependencies in the examples on the learning process. In Section 4 we use these results to calculate how the presentation of examples affects asymptotic performances like the representation error and the prediction error. The representation error measures how well the environment is represented by the network after learning. The prediction error is the average error that a continually learning network makes on the next example. In Section 5 we use the results of Section 3 to study the effect of dependencies when the learning process is stuck on a so-called plateau in the error surface. Plateaus are frequently present in the error surface of multilayer perceptrons (Hush *et al.* 1992). Using the results in this section, the remarkable difference between randomized learning and natural learning, which has been mentioned in the previous paragraph, is explained. The last section gives a brief summary and discussion.

2 The Framework

In many on-line learning processes the weight change at learning step n can be written in the general form

$$\Delta \boldsymbol{w}(n) \equiv \boldsymbol{w}(n+1) - \boldsymbol{w}(n) = \eta \boldsymbol{f}[\boldsymbol{w}(n), \boldsymbol{x}(n)] \tag{2.1}$$

with $\boldsymbol{w}(n)$ the network weights and $\boldsymbol{x}(n)$ the presented example at iteration step n. η is the learning parameter, which is assumed to be constant in this paper, and $\boldsymbol{f}(\cdot,\cdot)$ the learning rule. Examples satisfying equation 2.1 can be found in supervised learning such as backpropagation for multilayer perceptrons (Werbos 1974; Rumelhart *et al.* 1986), where the examples $\boldsymbol{x}(n)$ are combinations of input vectors $[x_1(n), \ldots, x_k(n)]$ and desired output vectors $[y_1(n), \ldots, y_l(n)]$, as well as in unsupervised learning such as Kohonen's self-organizing rule for topological feature maps (Kohonen 1982), where $\boldsymbol{x}(n)$ stands for the input vector $[x_1(n), \ldots, x_k(n)]$. On-line learning in the general form of equation 2.1 has been studied extensively (Amari 1967; Ritter and Schulten 1988; White 1989; Heskes and Kappen 1991; Leen and Moody 1992; Orr and Leen 1992; Hansen *et al.* 1993; Radons 1993; Finnoff 1994). Many papers on this subject have been restricted to independent presentation of examples; i.e., the probability

$p(\boldsymbol{x}, n)$ to present an example $\boldsymbol{x}$ at iteration step n is given by a probability distribution $\rho(\boldsymbol{x})$, independent of its predecessor. Dependencies between successive examples have been studied in Benviste *et al.* (1987, and references herein) and recently in Kuan and White (1994) and Wiegerinck and Heskes (1994).

In this paper dependencies between examples are incorporated by assuming that the probability to present an example $\boldsymbol{x}$ depends on its predecessor $\boldsymbol{x}'$ through a transition probability $\tau(\boldsymbol{x}|\boldsymbol{x}')$, i.e., that $p(\boldsymbol{x}, n)$ follow a first-order stationary Markov process

$$p(\boldsymbol{x}, n+1) = \int d\boldsymbol{x}' \tau(\boldsymbol{x}|\boldsymbol{x}') p(\boldsymbol{x}', n). \tag{2.2}$$

Learning with independent examples is a special case with $\tau(\boldsymbol{x}|\boldsymbol{x}') = \rho(\boldsymbol{x})$. The limitation to first-order Markov processes is not as severe as it might seem at first sight, since stationary Markov processes of any finite order k can be incorporated in the formalism by redefining the vectors $\boldsymbol{x}$ to include the last k examples (Wiegerinck and Heskes 1994). The Markov process is assumed to have a unique asymptotic or stationary distribution $\rho(\boldsymbol{x})$, i.e., we assume that we can take limits like

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} o\left[\boldsymbol{x}(n)\right] = \int d\boldsymbol{x} \rho(\boldsymbol{x}) o(\boldsymbol{x})$$

in which $o(\boldsymbol{x})$ is some function of the patterns. So $\rho(\boldsymbol{x})$ describes the (asymptotic) relative frequency of patterns. A randomized learning strategy therefore will select its independent examples from this stationary distribution. In this paper we will denote these long time averages with brackets $\langle\cdot\rangle_x$.

$$\langle o(\boldsymbol{x})\rangle_x \equiv \int d\boldsymbol{x} \rho(\boldsymbol{x}) o(\boldsymbol{x})$$

and sometimes we use capitals, i.e., we define quantities like $\Phi \equiv \langle \phi(\boldsymbol{x})\rangle_x$.

Many neural network algorithms, including backpropagation, perform gradient descent on a "local" cost or error function $e(\boldsymbol{w}, \boldsymbol{x})$,

$$\boldsymbol{f}(\boldsymbol{w}(n), \boldsymbol{x}(n)) \equiv -\nabla_w e(\boldsymbol{w}(n), \boldsymbol{x}(n)). \tag{2.3}$$

The idea of this learning rule is that with a small learning parameter, the stochastic gradient descent (equations 2.1 and 2.3) approximates deterministic gradient descent on the "global" error potential

$$E(\boldsymbol{w}) \equiv \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} e\left[\boldsymbol{w}, \boldsymbol{x}(n)\right]. \tag{2.4}$$

We restrict ourselves to learning with a cost function in order to compare performances between several types of pattern presentation (with equal stationary distributions), in particular in Sections 4 and 5. However, most derivations and results in this paper can be easily generalized to the general rule in equation 2.1.

3 ODE Approximation and Beyond

The update rule for the weights in equation 2.1 and the Markov process governing the presentation of examples in equation 2.2 can be combined into one evolution equation for the joint probability $\hat{P}(\boldsymbol{w}, \boldsymbol{x}, n)$ that at step n example $\boldsymbol{x}$ is presented to the network with weight vector $\boldsymbol{w}$. This probability obeys the Markov process

$$\hat{P}(\boldsymbol{w}, \boldsymbol{x}, n+1) = \int d\boldsymbol{w}'\, d\boldsymbol{x}'\, \tau(\boldsymbol{x}|\boldsymbol{x}')\, \delta(\boldsymbol{w} - \boldsymbol{w}' - \eta \boldsymbol{f}(\boldsymbol{w}', \boldsymbol{x}'))\, \hat{P}(\boldsymbol{w}', \boldsymbol{x}', n). \quad (3.1)$$

We are interested in the learning process, i.e., in the evolution of the probability distribution of weights

$$P(\boldsymbol{w}, n) = \int d\boldsymbol{x}\, \hat{P}(\boldsymbol{w}, \boldsymbol{x}, n).$$

With dependent examples, it is not possible to derive a self-supporting equation for the evolution of $P(\boldsymbol{w}, n)$ by direct integration over $\boldsymbol{x}$ in equation 3.1. However, in Weigerinck and Heskes (1994) it is shown that the evolution equation of $P(\boldsymbol{w}, n)$ can be expanded systematically in the small learning parameter η. The basic assumption for this expansion is that the dynamics of the weights, with typical time scale $1/\eta$, is much slower than the typical time scale of the examples.

In the following, we present a slightly different approach to approximate the evolution of the probability distribution of weights. This approach, based on van Kampen (1992), assumes that the distribution of weights, with initial form $P(\boldsymbol{w}, 0) = \delta[\boldsymbol{w} - \boldsymbol{w}(0)]$, remains sharply peaked as n increases. We follow the heuristic treatment in Benviste *et al.* (1987) and average the learning rule over a "mesoscopic" time scale (Hansen *et al.* 1993), which is much larger than the typical time scale of the example dynamics yet much smaller than the time scale on which the weights can change significantly. With the averaged learning rule we can directly calculate approximate equations for the mean $\overline{\boldsymbol{w}}(n)$ and the covariance matrix $\Sigma^2(n)$, which describe the position and the width of the peak $P(\boldsymbol{w}, n)$, respectively.

We iterate the learning step from equation 2.1 M times, where M is a mesoscopic time scale, i.e., $1 \ll M \ll 1/\eta$, and obtain

$$\boldsymbol{w}(n+M) - \boldsymbol{w}(n) = \eta \sum_{m=0}^{M-1} \boldsymbol{f}[\boldsymbol{w}(n+m), \boldsymbol{x}(n+m)]. \quad (3.2)$$

For the average $\overline{\boldsymbol{w}}(n) \equiv \langle \boldsymbol{w}(n) \rangle$ (brackets $\langle \ldots \rangle$ stand for averaging over the combined process in equation 3.1), we have the exact identity

$$\overline{\boldsymbol{w}}(n+M) - \overline{\boldsymbol{w}}(n) = \eta \sum_{m=0}^{M-1} \langle \boldsymbol{f}[\boldsymbol{w}(n+m), \boldsymbol{x}(n+m)] \rangle. \quad (3.3)$$

On the one hand, the mesoscopic time scale is much smaller than the time scale on which the probability distribution $P(\boldsymbol{w}, n)$ can change appreciably. Therefore, if the probability distribution $P(\boldsymbol{w}, n)$ is very sharply peaked, we can expand equation 3.3 around the mean $\overline{\boldsymbol{w}}(n)$

$$\overline{\boldsymbol{w}}(n+M) - \overline{\boldsymbol{w}}(n) = \eta \sum_{m=0}^{M-1} \langle \boldsymbol{f}\left[\overline{\boldsymbol{w}}(n), \boldsymbol{x}(n+m)\right]\rangle + \cdots .$$

On the other hand, the mesoscopic time scale is much larger than the typical time scale of the Markov process governing the presentation of examples. Therefore we can approximate the sum

$$\begin{aligned} \frac{1}{M}\sum_{m=0}^{M-1} \langle \boldsymbol{f}\left[\overline{\boldsymbol{w}}(n), \boldsymbol{x}(n+m)\right]\rangle &\approx \lim_{N\to\infty} \frac{1}{N}\sum_{m=0}^{N-1} \langle \boldsymbol{f}\left[\overline{\boldsymbol{w}}(n), \boldsymbol{x}(n+m)\right]\rangle \\ &\equiv \boldsymbol{F}\left[\overline{\boldsymbol{w}}(n)\right]. \end{aligned} \tag{3.4}$$

Thus, in lowest order, the stochastic equation 3.3 can be approximated by the deterministic difference equation

$$\overline{\boldsymbol{w}}(n+M) - \overline{\boldsymbol{w}}(n) = \eta M \boldsymbol{F}\left[\overline{\boldsymbol{w}}(n)\right].$$

For small ηM, the difference equation for the position of the peak turns into an ordinary differential equation (ODE). In terms of the rescaled continuous time t, with $t_n \equiv \eta n$ [we will use both notations $\boldsymbol{w}(n)$ and $\boldsymbol{w}(t)$], we obtain that the learning process is approximated by the ODE

$$\frac{d\overline{\boldsymbol{w}}(t)}{dt} = \boldsymbol{F}\left[\overline{\boldsymbol{w}}(t)\right] = -\nabla E\left[\overline{\boldsymbol{w}}(t)\right]. \tag{3.5}$$

In this equation $E(\boldsymbol{w})$ is the global error potential defined in equation 2.4. In lowest order the weights do indeed follow the gradient of the global error potential. Dependencies in successively presented examples have no influence on the ODE (equation 3.5): this equation depends only on the stationary distribution $\rho(\boldsymbol{x})$ of the examples. Corrections to the ODE arise when we expand (equation 3.2)

$$\begin{aligned} \boldsymbol{w}(n+M) - \boldsymbol{w}(n) &= \eta \sum_{m=0}^{M-1} \boldsymbol{f}\left[\boldsymbol{w}(n+m), \boldsymbol{x}(n+m)\right] \\ &= \eta \sum_{m=0}^{M-1} \boldsymbol{f}\left[\boldsymbol{w}(n), \boldsymbol{x}(n+m)\right] \\ &\quad + \eta \sum_{m=0}^{M-1} \mathsf{h}\left[\boldsymbol{w}(n), \boldsymbol{x}(n+m)\right] \\ &\quad \times \left[\boldsymbol{w}(n+m) - \boldsymbol{w}(n)\right] + \cdots \\ &= \eta \sum_{m=0}^{M-1} \boldsymbol{f}\left[\boldsymbol{w}(n), \boldsymbol{x}(n+m)\right] \end{aligned}$$

$$+ \eta^2 \sum_{m=0}^{M-1} \mathsf{h}\left[\boldsymbol{w}(n), \boldsymbol{x}(n+m)\right] \times \sum_{l=0}^{m-1} \boldsymbol{f}\left[\boldsymbol{w}(n), \boldsymbol{x}(n+l)\right] + \cdots \tag{3.6}$$

with the "local Hessian" $\mathsf{h}(\boldsymbol{w}, \boldsymbol{x}) \equiv \boldsymbol{\nabla}_w \boldsymbol{\nabla}_w^T e(\boldsymbol{w}, \boldsymbol{x})$. Using the separation between time scales, we approximate this expansion by

$$\boldsymbol{w}(n+M) - \boldsymbol{w}(n) = \eta M \left\{ \boldsymbol{F}\left[\boldsymbol{w}(n)\right] + \eta \boldsymbol{B}\left[\boldsymbol{w}(n)\right] - \frac{1}{2}\eta M \mathsf{H}\left[\boldsymbol{w}(n)\right] \boldsymbol{F}(\boldsymbol{w}(n) + \cdots \right\} \tag{3.7}$$

with the "Hessian"

$$\mathsf{H}(\boldsymbol{w}) \equiv \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \left\langle \mathsf{h}\left[\boldsymbol{w}, \boldsymbol{x}(n)\right] \right\rangle_x \tag{3.8}$$

and

$$\begin{aligned} \boldsymbol{B}(\boldsymbol{w}) &\equiv \lim_{N\to\infty} \frac{1}{N} \sum_{m=0}^{N-1} \left\langle \left\{ \mathsf{h}\left[\boldsymbol{w}, \boldsymbol{x}(m)\right] - \mathsf{H}(\boldsymbol{w}) \right\} \sum_{l=0}^{m-1} \left\{ \boldsymbol{f}\left[\boldsymbol{w}, \boldsymbol{x}(l)\right] - \boldsymbol{F}(\boldsymbol{w}) \right\} \right\rangle_x \\ &= \lim_{N\to\infty} \sum_{n=1}^{N-1} \left[1 - \frac{n}{N}\right] \\ &\quad \times \left\langle \left\{ \mathsf{h}\left[\boldsymbol{w}, \boldsymbol{x}(n)\right] - \mathsf{H}(\boldsymbol{w}) \right\} \left\{ \boldsymbol{f}\left[\boldsymbol{w}, \boldsymbol{x}(0)\right] - \boldsymbol{F}(\boldsymbol{w}) \right\} \right\rangle_x . \end{aligned} \tag{3.9}$$

Note that $\boldsymbol{B}(\boldsymbol{w})$ is zero with independent examples. Later on we will see that the term containing $\mathsf{H}\left[\boldsymbol{w}(n)\right] \boldsymbol{F}\left[\boldsymbol{w}(n)\right]$ will vanish by the transformation to continuous time.

Averaging equation 3.7 yields

$$\overline{\boldsymbol{w}}(n+M) - \overline{\boldsymbol{w}}(n) = \eta M \left\{ \left\langle \boldsymbol{F}\left[\boldsymbol{w}(n)\right] \right\rangle + \eta \left\langle \boldsymbol{B}\left[\boldsymbol{w}(n)\right] \right\rangle - \frac{1}{2}\eta M \left\langle \mathsf{H}\left[\boldsymbol{w}(n)\right] \boldsymbol{F}\left[\boldsymbol{w}(n)\right] \right\rangle + \cdots \right\}$$

and by expansion of the right-hand side around the mean $\overline{\boldsymbol{w}}(n)$ we obtain

$$\begin{aligned} &\overline{\boldsymbol{w}}(n+M) - \overline{\boldsymbol{w}}(n) \\ &= \eta M \left\{ \boldsymbol{F}\left[\overline{\boldsymbol{w}}(n)\right] - \frac{1}{2} \mathsf{Q}\left[\overline{\boldsymbol{w}}(n)\right] : \left\langle \left[\boldsymbol{w}(n) - \overline{\boldsymbol{w}}(n)\right] \left[\boldsymbol{w}(n) - \overline{\boldsymbol{w}}(n)\right]^T \right\rangle \right. \\ &\quad \left. + \eta \boldsymbol{B}\left[\overline{\boldsymbol{w}}(n)\right] - \frac{1}{2}\eta M \mathsf{H}\left[\overline{\boldsymbol{w}}(n)\right] \boldsymbol{F}\left[\overline{\boldsymbol{w}}(n)\right] + \cdots \right\} \\ &= \eta M \left\{ \boldsymbol{F}\left[\overline{\boldsymbol{w}}(n)\right] - \frac{1}{2} \mathsf{Q}\left[\overline{\boldsymbol{w}}(n)\right] : \Sigma^2(n) + \eta \boldsymbol{B}\left[\overline{\boldsymbol{w}}(n)\right] \right. \\ &\quad \left. - \frac{1}{2}\eta M \mathsf{H}\left[\overline{\boldsymbol{w}}(n)\right] \boldsymbol{F}\left[\overline{\boldsymbol{w}}(n)\right] + \cdots \right\} \end{aligned}$$

in which

$$\Sigma^2(n) \equiv \left\langle [\boldsymbol{w}(n) - \overline{\boldsymbol{w}}(n)][\boldsymbol{w}(n) - \overline{\boldsymbol{w}}(n)]^T \right\rangle \tag{3.10}$$

$$Q_{\alpha\beta\gamma}(\boldsymbol{w}) \equiv -\frac{\partial^2 F_\alpha(\boldsymbol{w})}{\partial w_\beta \partial w_\gamma} \tag{3.11}$$

$$(\mathsf{Q}:\Sigma^2)_\alpha \equiv \sum_{\beta\gamma} Q_{\alpha\beta\gamma}\Sigma^2_{\beta\gamma}. \tag{3.12}$$

Transformation to continuous time finally yields a first approximation beyond the ODE in equation 3.5

$$\frac{d\overline{\boldsymbol{w}}(t)}{dt} = \boldsymbol{F}\left[\overline{\boldsymbol{w}}(t)\right] - \frac{1}{2}\mathsf{Q}\left[\overline{\boldsymbol{w}}(t)\right] : \Sigma^2(t) - \eta \boldsymbol{B}\left[\overline{\boldsymbol{w}}(t)\right]. \tag{3.13}$$

Unlike equation 3.5 this is no longer a self-supporting equation for $\overline{\boldsymbol{w}}$ alone; higher moments enter as well. The evolution of the mean $\overline{\boldsymbol{w}}$ in the course of time is therefore not determined by $\overline{\boldsymbol{w}}$ itself, but is influenced by the fluctuations around this average through their covariance Σ^2. It is clear that for the existence of the ODE approximation and of its higher order approximations, it is neccessary that the fluctuations are small. In derivation of equation 3.13 we have used in foresight that these fluctuations are of order $\sqrt{\eta}$ and therefore their covariance Σ^2 is of order η. In fact, similarly to the derivations of equations 3.5 and 3.13, a lowest order approximation for the fluctuations can be derived,

$$\frac{d\Sigma^2(t)}{dt} = -\mathsf{H}\left[\overline{\boldsymbol{w}}(t)\right]\Sigma^2(t) - \Sigma^2(t)\mathsf{H}\left[\overline{\boldsymbol{w}}(t)\right] + \eta \mathsf{D}\left[\overline{\boldsymbol{w}}(t)\right] \tag{3.14}$$

with the "diffusion" matrix

$$\mathsf{D}(\boldsymbol{w}) = \left\langle \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} \{\boldsymbol{f}\left[\boldsymbol{w}, \boldsymbol{x}(n)\right] - \boldsymbol{F}(\boldsymbol{w})\}\{\boldsymbol{f}\left[\boldsymbol{w}, \boldsymbol{x}(m)\right] - \boldsymbol{F}(\boldsymbol{w})\}^T \right\rangle_x. \tag{3.15}$$

From equation 3.14, we can see that $\Sigma^2(t)$ remains bounded if H is positive definite. In this case $\Sigma^2(t) = \mathcal{O}(\eta)$, which makes equation 3.14 with equation 3.13 a valid approximation (van Kampen 1992). In other words, since η is small, this justifies a posteriori the assumption that $P(\boldsymbol{w}, n)$ is sharply peaked. In other cases where the fluctuations do not remain bounded, the approximation is applicable only during a short period.

The diffusion $\mathsf{D}(\boldsymbol{w})$ can be expressed as the sum of an independent and a dependent part:

$$\begin{aligned} \mathsf{D}(\boldsymbol{w}) &= \mathsf{C}_0(\boldsymbol{w}) + \lim_{N\to\infty} \sum_{n=1}^{N-1} \left[1 - \frac{n}{N}\right] \left[\mathsf{C}_n(\boldsymbol{w}) + \mathsf{C}_n^T(\boldsymbol{w})\right] \\ &\equiv \mathsf{C}_0(\boldsymbol{w}) + \mathsf{C}_+(\boldsymbol{w}) \end{aligned} \tag{3.16}$$

where we have defined the auto-correlation matrices

$$\mathsf{C}_n(\boldsymbol{w}) \equiv \left\langle \{\boldsymbol{f}\left[\boldsymbol{w}, \boldsymbol{x}(n)\right] - \boldsymbol{F}(\boldsymbol{w})\}\{\boldsymbol{f}\left[\boldsymbol{w}, \boldsymbol{x}(0)\right] - \boldsymbol{F}(\boldsymbol{w})\}^T \right\rangle_x. \tag{3.17}$$

For on-line learning with random sampling, there are no dependencies between subsequent weight changes, so $\mathsf{C}_+(\boldsymbol{w}) = 0$ and consequently the diffusion $\mathsf{D}(\boldsymbol{w})$ reduces to $\mathsf{C}_0(\boldsymbol{w})$ (see, e.g., Heskes 1994).

The set of equations 3.13 and 3.14 for $\overline{\boldsymbol{w}}$ and Σ^2 forms a self-supporting first approximation beyond the ODE approximation in equation 3.5. It is not necessary to solve equations 3.13 and 3.14 simultaneously. Since the covariance Σ^2 appears in equation 3.13 as a correction it suffices to compute Σ^2 from equation 3.14 using the ODE approximation for $\overline{\boldsymbol{w}}$. Following van Kampen (1992) we set $\overline{\boldsymbol{w}} = \boldsymbol{w}_{\text{ODE}} + \boldsymbol{u}$, and solve

$$\frac{d\boldsymbol{w}_{\text{ODE}}(t)}{dt} = \boldsymbol{F}\left[\boldsymbol{w}_{\text{ODE}}(t)\right] \tag{3.18}$$

$$\begin{aligned}\frac{d\Sigma^2(t)}{dt} &= -\mathsf{H}\left[\boldsymbol{w}_{\text{ODE}}(t)\right]\Sigma^2(t) - \Sigma^2(t)\mathsf{H}\left[\boldsymbol{w}_{\text{ODE}}(t)\right] \\ &\quad + \eta\mathsf{D}\left[\boldsymbol{w}_{\text{ODE}}(t)\right]\end{aligned} \tag{3.19}$$

$$\begin{aligned}\frac{d\boldsymbol{u}(t)}{dt} &= -\mathsf{H}\left[\boldsymbol{w}_{\text{ODE}}(t)\right]\boldsymbol{u}(t) - \frac{1}{2}\mathsf{Q}\left[\boldsymbol{w}_{\text{ODE}}(t)\right] : \Sigma^2(t) \\ &\quad -\eta\boldsymbol{B}\left[\boldsymbol{w}_{\text{ODE}}(t)\right].\end{aligned} \tag{3.20}$$

Equations 3.18 and 3.19 are equivalent to results that one can find in the literature (Beneviste *et al.* 1987; Kuan and White 1994; Wiegerinck and Heskes 1994). The ODE in equation 3.18 approximates in lowest order the dynamics of the weights. The covariance matrix $\Sigma^2(t)$, which obeys equation 3.19, describes the stochastic deviations $\boldsymbol{w}(n) - \boldsymbol{w}_{\text{ODE}}(t_n)$ between the weights and the ODE approximation. These fluctuations are typically of order $\sqrt{\eta}$. [Their "square" $\Sigma^2(t)$ is of order η.] In Benviste *et al.* (1987) and Kuan and White (1994) a Wiener process is rigorously derived to describe these fluctuations. In Wiegerinck and Heskes (1994) a Fokker-Planck equation that describes these fluctuations is derived. In the next section we will study how these fluctuations affect some asymptotic error measures.

Equation 3.20 decribes a bias $\boldsymbol{u}$ between the mean $\overline{\boldsymbol{w}}$ and the ODE approximation $\boldsymbol{w}_{\text{ODE}}$. The dynamics of the bias consists of two driving terms. The first one is the interaction between the nonlinearity of the learning rule Q and the fluctuations described by Σ^2. This term can be understood in the following way: If a random fluctuation into one direction in weight space does not result in the same restoring effect as a random fluctuation into the opposite direction, then random fluctuations will obviously result in a netto bias effect. The other driving term in equation 3.20 is $\boldsymbol{B}$ (see equation 3.9). This term is only due to the dependencies of the examples. Since the two driving terms are typically of order η, the bias term is also typically of order η. The bias is typically an order $\sqrt{\eta}$ smaller than the fluctuations and therefore neglected in regular situations. However, in Section 5 it will be shown (and this will be supported by simulations) that there are situations where this bias term is of crucial importance.

As an approximation, the set of coupled equations 3.18–3.20 is equally valid as the coupled set 3.13 and 3.14. However, in 3.18–3.20 the hierarchical structure of the approximations (ODE approximation, fluctuations, bias) is clearer.

The influence of the example presentation on the evolution of the weight distribution is twofold. On the one side, dependencies between examples affect the covariance Σ^2 through the diffusion term D (see equation 3.15). On the other side, they affect the mean value through the vector $\boldsymbol{B}$, and indirectly through the covariance Σ^2. For independent examples, D reduces to C_0 (see equation 3.17) and $\boldsymbol{B} = 0$ exactly.

Finally we want to stress that the essential assumption for the validity of equations 3.18–3.20 is that the weights can be described by their average value with small superimposed fluctuations. In other words, the approximation 3.18–3.20 is locally valid. This is the case if the Hessian H is positively definite. In other cases the approximation is valid only for short times (van Kampen 1992). In the analysis of the next two sections we tacitly assume this local validity.

4 Representation Error and Prediction Error

In this section we show how dependencies between successive examples influence the asymptotic performance of the network. In the asymptotic situation, the weights are assumed to remain concentrated around a minimum w^* of the global error $E(w)$. We consider two measures of network performance: the "representation error" E_{repr} and the "prediction error" E_{pred}. The meaning of this terminology differs slightly from its usual meaning in most neural network literature. The representation error

$$E_{\text{repr}} \equiv \lim_{n\to\infty} \langle \langle e[w(n), x] \rangle_x \rangle = \lim_{n\to\infty} \langle E[w(n)] \rangle \tag{4.1}$$

is the expectation of the asymptotic global error $E[w(\infty)]$ (cf. equation 2.4) made by the network. It is a useful measure to compare different example presentation techniques if the goal is minimization of the local cost function $e(w, x)$ in an environment given by a probability distribution $\rho(x)$. In the context of time series, E_{repr} measures how well the asymptotic network state is expected to represent the whole time series. The prediction error

$$E_{\text{pred}} \equiv \lim_{n\to\infty} \langle e[w(n), x(n)] \rangle \tag{4.2}$$

is the error that the network in its final stage of learning is expected to make on the *next* example of the time-series. E_{pred} measures the error locally in time, in contrast to the more global measure E_{repr}.

The weights are assumed to be concentrated around a minimum w^* of the global error $E(w)$. This implies

$$\nabla E(w^*) = 0.$$

The fluctuations Σ^2 and the bias $\boldsymbol{u} = \langle \boldsymbol{w}(\infty) \rangle - \boldsymbol{w}^*$ satisfy in lowest order the fixed point equations of 3.19 and 3.20

$$\begin{aligned} \mathsf{H}(\boldsymbol{w}^*)\Sigma^2 + \Sigma^2\mathsf{H}(\boldsymbol{w}^*) &= \eta \mathsf{D}(\boldsymbol{w}^*) \\ \mathsf{H}(\boldsymbol{w}^*)\boldsymbol{u} &= -\frac{1}{2}\mathsf{Q}(\boldsymbol{w}^*) : \Sigma^2 - \eta \boldsymbol{B}(\boldsymbol{w}^*). \end{aligned}$$

With the techniques used in the previous section we calculate the two error measures up to $\mathcal{O}(\eta)$. To obtain the representation error in equation 4.1 we expand $E(\boldsymbol{w})$ around its minimum $\boldsymbol{w}^*$,

$$\begin{aligned} E_{\text{repr}} = \lim_{n\to\infty} \langle E[\boldsymbol{w}(n)] \rangle &= E(\boldsymbol{w}^*) + \frac{1}{2}\operatorname{Tr}\left[\mathsf{H}(\boldsymbol{w}^*)\Sigma^2\right] + \cdots \\ &= E(\boldsymbol{w}^*) + \frac{\eta}{4}\operatorname{Tr}\left[\mathsf{D}(\boldsymbol{w}^*)\right] + \cdots . \end{aligned}$$

To calculate the prediction error in equation 4.2, we apply a time-averaging procedure similar to the one used in Section 3. Given weight vector $\boldsymbol{w}(n)$ before the first learning step, the local error over the next M examples is

$$\begin{aligned} \frac{1}{M}\sum_{m=0}^{M-1} e[\boldsymbol{w}(n+m), \boldsymbol{x}(n+m)] &= \frac{1}{M}\sum_{m=0}^{M-1} e[\boldsymbol{w}(n), x(n+m)] \\ &\quad - \frac{\eta}{M}\sum_{m=0}^{M-1}\sum_{l=0}^{m-1} \boldsymbol{f}^T[\boldsymbol{w}(n), \boldsymbol{x}(n+m)] \\ &\quad \times \boldsymbol{f}[\boldsymbol{w}(n), \boldsymbol{x}(n+l)] + \cdots . \end{aligned}$$

For a mesoscopic timescale M we obtain, using the definitions from equations 3.16 and 3.17,

$$\begin{aligned} E_{\text{pred}} &= \lim_{n\to\infty} \frac{1}{M}\sum_{m=0}^{M-1} \langle e[\boldsymbol{w}(n+m), \boldsymbol{x}(n+m)] \rangle \\ &= E_{\text{repr}} - \frac{\eta}{2}\operatorname{Tr}[\mathsf{C}_+(\boldsymbol{w}^*)] + \cdots . \end{aligned}$$

For randomized learning, the prediction error and representation error are equal: $E_{\text{pred}} = E_{\text{repr}} \equiv E_{\text{ran}}$. Using $\mathsf{D}(\boldsymbol{w}^*) = \mathsf{C}_0(\boldsymbol{w}^*)$, we obtain

$$E_{\text{ran}} = E(\boldsymbol{w}^*) + \frac{\eta}{4}\operatorname{Tr}[\mathsf{C}_0(\boldsymbol{w}^*)] + \cdots .$$

If we compare the representation and prediction error with dependent examples to the error with independent examples (assuming that the weight distribution is concentrated around the same minimum $\boldsymbol{w}^*$), we see that, up to order η, the profit in prediction exactly cancels the loss in representation and vice versa:

$$\frac{E_{\text{pred}} + E_{\text{repr}}}{2} = E_{\text{ran}} + \cdots .$$

In the context of strategies to select examples, this implies that a strategy that yields a larger prediction error will most likely lead to a smaller representation error. Depending on whether successive weight changes are, roughly speaking, positively or negatively correlated, the prediction error is smaller or larger than the representation error, respectively. This is nicely illustrated by the following simple example.

We consider a process where the examples can take two values $x = \pm 1$ with transition probability

$$\tau(x|x') = (1-q)\delta_{x,x'} + q\delta_{x,-x'},$$

i.e., there is a probability $0 < q \leq 1$ to flip the sign of the input. The stationary distribution $\rho(x)$ is given by

$$\rho(x) = \frac{1}{2}(\delta_{x,-1} + \delta_{x,1}). \tag{4.3}$$

A one-dimensional "weight" w tries to minimize the squared distance between the presented example and the weight; i.e., the local error is

$$e(w,x) = \frac{1}{2}(w-x)^2 \tag{4.4}$$

and the corresponding update rule [cf. equations 2.1 and 2.3] is

$$\Delta w = \eta(x-w).$$

The global error $E(w)$ (cf. equation 2.4) is obtained by averaging the local error (cf. equation 4.4) over the stationary distribution (cf. equation 4.3),

$$E(w) = \frac{1}{4}(1-w)^2 + \frac{1}{4}(-1-w)^2 = \frac{1}{2} + \frac{1}{2}w^2$$

and has a minimum $E(w^*) = 1/2$ for $w^* = 0$. To compute the performance measures from equations 4.1 and 4.2 for our simple unsupervised example as a function of the flip probability q, we first calculate the autocorrelations $C_m(w^*)$ (cf. equation 3.17) in the minimum $w^* = 0$:

$$C_m(0) = \langle x(m)x(0)\rangle_x = (1-2q)^m \qquad \text{and thus} \qquad C_0 = 1$$

$$\text{and} \qquad C_+ = \frac{1-2q}{q}.$$

Up to $\mathcal{O}(\eta)$ we obtain

$$E_{\text{pred}} = \frac{1}{2} + \frac{3q-1}{4q}\eta \qquad \text{and} \qquad E_{\text{repr}} = \frac{1}{2} + \frac{1-q}{4q}\eta.$$

For flip probability $q < 1/2$ we have better prediction than representation; for $q > 1/2$ better representation than prediction ($q = 1/2$ corresponds to randomized learning). This is what we could expect: The larger the flip probability, the better the overall sampling of the input space for the problem at hand (finding the average input) and thus the better the representation. However, the larger the flip probability, the more difficult to predict the next example for the network that has just been adapted to the current example.

5 Plateaus

In comparing the asymptotic performance of networks trained with dependent and independent examples in the previous section, we assumed that with small η, both types of learning lead to the same (local) minimum of the global error $E(\boldsymbol{w})$ (see equation 2.4). This is not unreasonable if the learning process is initiated in the neighborhood of this minimum. A minimum is a stable equilibrium point of the ODE dynamics (cf. equation 3.5), i.e, the eigenvalues of the Hessian $\mathsf{H}(\boldsymbol{w})$ (see equation 3.8) are strictly positive. In the neighborhood of a minimum, the ODE force $\boldsymbol{F}(\boldsymbol{w})$ (see equation 3.4) is the dominating factor in the dynamics. Perturbations due to the higher order corrections are immediately restored by the ODE force.

In this section, however, we will consider so-called "plateaus." Plateaus are flat spots on the global-error surface. They are often the cause of the extremely long learning times and/or the bad convergence results in multilayer perceptron applications with the backpropagation algorithm (Hush *et al.* 1992). On a plateau, the gradient of E is negligible and H has some positive eigenvalues but also some zero eigenvalues. Plateaus can be viewed as indifferent equilibrium points of the ODE dynamics. Even with small η, the higher order terms can make the weight vector move around in the subspace of eigenvectors of H with zero eigenvalue without being restored by $\boldsymbol{F}$. In other words, in these directions the higher order terms—in the first place the fluctuations, which are of order $\sqrt{\eta}$, and in the second place the bias, which is of order η—may give a larger contribution to the dynamics than the ODE term. Since the higher order terms are related to dependencies between the examples, on plateaus the presentation order of examples might significantly affect the learning process.

The effect of different example presentations in learning on a plateau will be illustrated by the following example. We consider the tent map

$$y(x) = 2(1/2 - |x - 1/2|), \qquad 0 \leq x \leq 1$$

which we view as a dynamic system producing a chaotic time series $x(n+1) = y[x(n)]$ (Schuster 1989). To model this system we use a two-layered perceptron with one input unit, two hidden units, and a linear output,

$$z(\boldsymbol{w}, x) = v_0 + \sum_{\beta=1}^{2} v_\beta \tanh(w_{\beta 1}.x + w_{\beta 0}).$$

We train the network with input-output pairs $\boldsymbol{x} = \{x, y(x)\}$ by on-line backpropagation (Rumelhart *et al.* 1986)

$$\Delta \boldsymbol{w} = -\eta \nabla_w e(\boldsymbol{w}, \boldsymbol{x})$$

with the usual squared error cost function

$$e(\boldsymbol{w}, \boldsymbol{x}) = [y(x) - z(\boldsymbol{w}, x)]^2/2.$$

We compare two types of example presentation. With natural learning, examples are presented according to the sequence generated by the tent map, i.e.

$$\begin{aligned} \boldsymbol{x}(0) &= \{x(0), y(x(0))\}, \\ \boldsymbol{x}(1) &= \{x(1), y(x(1))\}, \ldots, \boldsymbol{x}(n) = \{x(n), y(x(n))\} \ldots . \end{aligned}$$

with $x(n+1) = y[x(n)]$ [and $x(0)$ randomly drawn from the interval $[0, 1]$]. With randomized learning, at each iteration step an input x is drawn according to the stationary distribution $\rho(x)$, i.e., homogeneously from the interval $[0, 1]$ (Schuster 1989), the corresponding output $y(x)$ is computed, and the pair $\{x, y(x)\}$ is presented to the network. In both cases we initialize with the same small weights, $-\epsilon < v_\gamma, w_{j\alpha} < \epsilon$. Small random weights are often recommended to prevent early saturation of the weights (Lee *et al.* 1991).

As reported earlier (Hondou and Sawada 1994), simulations show a dramatic difference between the two learning strategies in their performance learning the tent map (cf. Figs. 1 and 2). To understand this difference, we will study the weight dynamics by local linearizations. In the neighborhood of a point $\boldsymbol{w}^*$ in weight space the ODE from equation 3.5 can be approximated by

$$\frac{d\overline{\boldsymbol{w}}(t)}{dt} = \boldsymbol{F}(\boldsymbol{w}^*) - \mathsf{H}(\boldsymbol{w}^*)[\overline{\boldsymbol{w}}(t) - \boldsymbol{w}^*]. \tag{5.1}$$

The weights are initialized at $\boldsymbol{w}(0) = \mathcal{O}(\epsilon)$, with $\epsilon \approx 0$. The linearization (cf. equation 5.1) around $\boldsymbol{w}^* = \boldsymbol{w}^{(0)} = 0$ yields an approximation of the weight dynamics during the initial stage of learning,

$$\frac{d}{dt}\begin{pmatrix} \overline{v}_0 \\ \overline{v}_j \\ \overline{w}_{j0} \\ \overline{w}_{j1} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{3} & -\frac{1}{6} \\ 0 & -\frac{1}{3} & 0 & 0 \\ 0 & -\frac{1}{6} & 0 & 0 \end{pmatrix} \begin{pmatrix} \overline{v}_0 \\ \overline{v}_j \\ \overline{w}_{j0} \\ \overline{w}_{j1} \end{pmatrix} \tag{5.2}$$

with $j = 1, 2$. From equation 5.2, we see that $\overline{v}_0$ quickly converges to $\overline{v}_0 = 1/2$ on a time scale where the other weights hardly change (cf. Fig. 3). In other words, during this stage the network just learns the average value of the target function. This is a well-known phenomenon: Backpropagation tends to select the gross structures of its environment first.

After the initial stage, equation 5.2 does not provide a good approximation any more. The linearization (cf. equation 5.1) of the ODE around *the new point* $\boldsymbol{w}^* = \boldsymbol{w}^{(1)} = (v_0^{(1)} = 1/2, v_j^{(1)} = 0, w_{j\alpha}^{(1)} = 0)$, (with $\alpha = 0, 1$ and $j = 1, 2$), describes the dynamics of the weights during the next stage,

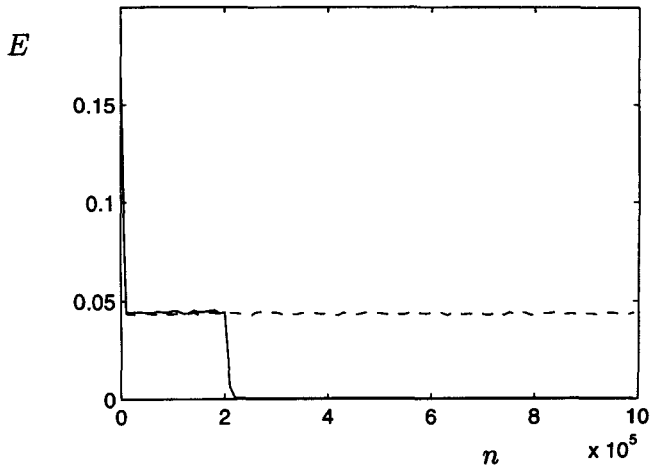

Figure 1: Typical global error E of natural learning (full curve) and of randomized learning (dashed curve). Simulation performed with a single network. Learning parameter $\eta = 0.1$. Weights initialization: $\epsilon = 10^{-4}$. Data points are plotted every 10^4 iterations.

$$\frac{d}{dt}\begin{pmatrix} \overline{v}_0 \\ \overline{v}_\beta \\ \overline{w}_{\beta 0} \\ \overline{w}_{\beta 1} \end{pmatrix} = -\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} \overline{v}_0 - \frac{1}{2} \\ \overline{v}_\beta \\ \overline{w}_{\beta 0} \\ \overline{w}_{\beta 1} \end{pmatrix}$$

with $\beta = 1, 2$. At this stage, $\boldsymbol{F} = 0$, while the Hessian H has one positive eigenvalue ($\lambda = 1$) and further only zero eigenvalues. In other words, at $\boldsymbol{w}^{(1)}$ the weights are stuck on a plateau.

To find out whether the weights can escape the plateau, we have to consider the contributions of the higher η corrections to the weight dynamics from equations 3.13 and 3.14. Linearization of this set of equations around $\boldsymbol{w}^{(1)}$ yields

$$\begin{aligned} \frac{d\overline{\boldsymbol{w}}(t)}{dt} &= -\mathsf{H}(\boldsymbol{w}^{(1)})[\overline{\boldsymbol{w}}(t) - \boldsymbol{w}^{(1)}] \\ &\quad -\frac{1}{2}\left\{\mathsf{Q}(\boldsymbol{w}^{(1)}) + \boldsymbol{\nabla}\mathsf{Q}(\boldsymbol{w}^{(1)})[\overline{\boldsymbol{w}}(t) - \boldsymbol{w}^{(1)}]\right\} : \Sigma^2(t) \\ &\quad -\eta\left\{\boldsymbol{B}(\boldsymbol{w}^{(1)}) + \boldsymbol{\nabla}\boldsymbol{B}(\boldsymbol{w}^{(1)})[\overline{\boldsymbol{w}}(t) - \boldsymbol{w}^{(1)}]\right\} \qquad (5.3) \\ \frac{d\Sigma^2(t)}{dt} &= -\mathsf{H}(\boldsymbol{w}^{(1)})\Sigma^2(t) - \Sigma^2(t)\mathsf{H}(\boldsymbol{w}^{(1)}) + \eta\mathsf{D}(\boldsymbol{w}^{(1)}). \qquad (5.4) \end{aligned}$$

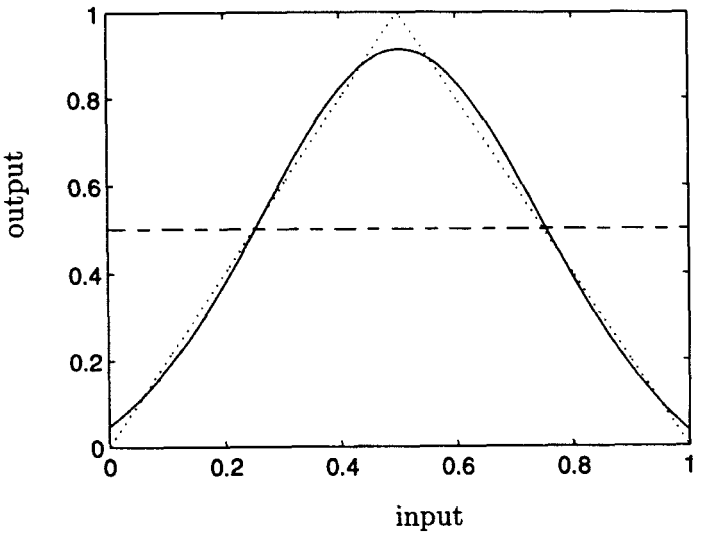

Figure 2: Typical network result after 10^6 iteration steps of natural learning (full curve) and randomized learning (dashed curve). The target function is the tent map (dotted curve). For simulation details, see caption of Figure 1.

At $w^{(1)}$, the (v_0, v_0) component is the only nonzero component for both the Hessian H and the diffusion D (for randomized learning as well as for natural learning). From equation 5.4, it thus follows that $\Sigma^2_{v_0,v_0}$ is the only nonzero component of the covariance matrix. So there will be fluctuations only in this direction. However, these fluctuations will be restored, due to the positive (v_0, v_0) component of the Hessian. Moreover, since $Q(w^{(1)})_{v_0 v_0 w}$ (see equation 3.11) and its derivatives vanish for all w, the covariance matrix Σ^2 does not couple with the (linearized) weight dynamics, and equation 5.3 reduces to the autonomous equation

$$\frac{d\overline{w}(t)}{dt} = -\mathsf{H}(w^{(1)})[\overline{w}(t) - w^{(1)}] - \eta \left\{ \boldsymbol{B}(w^{(1)}) + \nabla \boldsymbol{B}(w^{(1)})[\overline{w}(t) - w^{(1)}] \right\}.$$

With natural learning, straightforward calculations yield $\boldsymbol{B}(w^{(1)}) = 0$ and $\nabla \boldsymbol{B}(w^{(1)}) = 0$, except for the components

$$\nabla B_{v_0 v_0} = \frac{1}{216}, \qquad \nabla B_{v_0 w_{01}} = \frac{1}{18}, \qquad \nabla B_{w_{01} v_0} = \frac{1}{18},$$

$$\nabla B_{w_{01} w_{00}} = \frac{1}{108}, \qquad \nabla B_{w_{01} w_{01}} = \frac{1}{216}$$

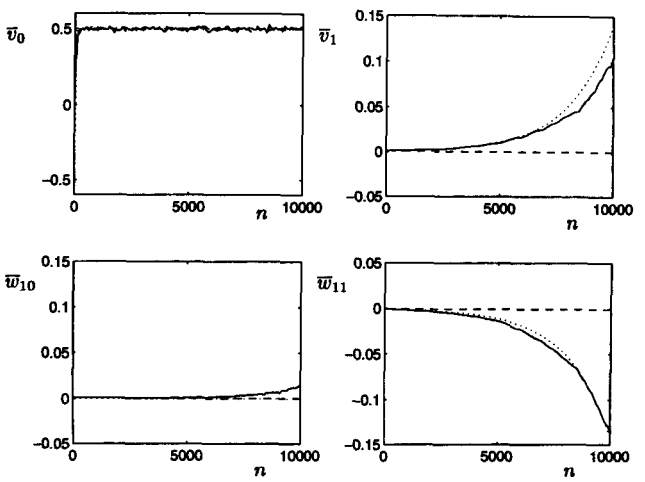

Figure 3: Weights obtained by simulations for natural learning (solid curves) and randomized learning (dashed curves) as functions of the number of iterations. Averaged over 100 iterations and an ensemble of 20 networks. The theoretical predictions computed with equation 5.5 are plotted as dotted curves.

with $\beta = 1, 2$. Concentrating on the dynamics of $\overline{v}_\beta$, and $\overline{w}_{\beta 1}$, we thus obtain the linear system

$$\frac{d}{dt}\begin{pmatrix} \overline{v}_\beta \\ \overline{w}_{\beta 1} \end{pmatrix} = -\frac{\eta}{216}\begin{pmatrix} 1 & 12 \\ 12 & 1 \end{pmatrix}\begin{pmatrix} \overline{v}_\beta \\ \overline{w}_{\beta 1} \end{pmatrix} \tag{5.5}$$

with $\beta = 1, 2$. This system has one negative eigenvalue λ_- and one positive eigenvalue $\lambda_+ = \eta\frac{11}{216}$. Along the direction of the eigenvector $(1, -1)$ corresponding to the positive eigenvalue, the weights will move away from $\boldsymbol{w}^{(1)}$ (cf. Fig. 3). Thus, natural learning escapes from the plateau, and reaches the global minimum (cf. Figs. 1 and 2). On the other hand, for randomized learning $\boldsymbol{B} = 0$ identically. This means that the weights of a randomized learning network are not helped by the higher η corrections and therefore cannot escape the plateau (cf. Figs. 1 and 2).

Figure 3 shows that the predictions computed with the theory agree well with the simulations of the neural network learning the tent map, and therefore we conclude that the difference in performance of the two learning strategies is well explained by the theory.

The analysis of this section—supported by the simulations—shows that if the learning process suffers from a plateau, then dependencies can help learning by a nonzero $\boldsymbol{B}$ term (cf. equation 3.9) with some posi-

tive eigenvalues. Of course, the magnitude of these eigenvalues and the direction of the corresponding eigenvectors depend strongly on the problem at hand, i.e., $\boldsymbol{B}$ is probably not for every problem directed toward a global minimum. But the fact remains that a nonzero $\boldsymbol{B}$ term can make the weights move *away* from the plateau, which facilitates an escape, resulting in a lower error. On the other hand, if a nonzero $\boldsymbol{B}$ does not make the weights wander away, or if it does not lead to an escape, the performance of dependent learning is still not worse than the performance of randomized learning, which also would get stuck on the plateau.

Another likely situation occurs if randomized learning *does* escape from the plateau, e.g., as a result of the fluctuations in the direction of a zero eigenvalue of the Hessian. In this situation the bias terms—which are an order $\sqrt{\eta}$ smaller than the fluctuations—can be neglected. In such a case dependent pattern presentation probably does not harm either—since similar fluctuations would also enhance escaping from the plateau with dependent patterns—unless the presentation order reduces the fluctuations too much! For instance, in the example of Section 4 the fluctuations are reduced if the examples are negatively correlated ($q > 1/2$). As a more realistic example, consider a problem with a fixed training set of P examples. A commonly used incremental learning strategy presents in each epoch of P learning steps each example only once (Haykin 1994). In other words, the patterns are aranged in a randomly ordered sequence $[\boldsymbol{x}(1), \ldots, \boldsymbol{x}(P)]$. It is obvious that this sequence-based or cyclic learning introduces dependencies between the examples. Moreover, the subsequent examples are negatively correlated. This follows from the fact that the probability to find identical subsequent examples is on average at least order P smaller in cyclic learning than in randomized learning. Indeed, it can be shown analytically that the leading term of the fluctuations completely vanishes in cyclic learning (Heskes and Wiegerinck 1996). As a consequence, randomized learning has a much larger chance to escape from a plateau than cyclic learning.

In conclusion, we recommend natural learning (with positive correlations) if the problem at hand suffers from a plateau. However, artificial dependencies introduced to reduce fluctuations are in such a case not advisable.

6 Summary and Discussion

This paper presents a quantitative analysis for on-line learning with dependent examples in a very general form. The analysis is based on two essential ingredients. One is the separation between the time scales of the example presentation and the weight dynamics. On the time scale needed for a representative sampling of the environment the weight changes must be negligible. A separation of time scales, which can be achieved using a small learning parameter, is essential in on-line learning to pre-

vent overspecialization on single examples. The other essential ingredient is the assumption that the weights can be described by their average value with small superimposed fluctuations. In other words, the theory is locally valid, and may therefore not be suited for quantitative computations of global properties of the learning process, such as the stationary distribution of weights or the escape time out of a local minimum. However, even a local theory can be useful to understand some aspects of global properties (Finnoff 1994). Our study of learning on plateaus is an example of a local analysis of on-line learning, which accounts for huge, nonlocal effects (Section 5).

In Section 3 we heuristically derived the first terms in a hierarchy of deterministic differential equations approximating the stochastic learning process. The leading term, the ODE term, only contains information of the stationary distribution of the examples. Dependencies between successive examples do not enter until the first correction to the ODE term. This implies that in general, when the ODE term is dominant, learning with dependent examples and learning with randomized examples are alike. The dependencies between examples merely act as corrections on the learning process, both in the fluctuations and in the bias. A rigorous derivation of the leading term, the ODE term, and of the Wiener process describing the fluctuations, can be found in Benviste *et al.* (1987) and Kuan and White (1994). To our knowledge, a rigorous derivation of higher order terms, such as the bias term in equation 3.20, has not been studied before.

In Section 4 we focused on the asymptotic convergence of the learning process in terms of representation error and prediction error. The representation error is the expected average error of the network with respect to the whole environment. It measures how well the environment is represented by the network after learning. The prediction error is the network's expected average error with respect to the next presented example. It can be viewed as a measure for the irregularity of the example presentation. A remarkable relation between representation and prediction error is that the more predictable the examples, the larger the representation error.

In Section 5 we studied on-line learning with a plateau. Plateaus are flat spots on the error surface that can severely slow down the learning process. In particular, backpropagation for multilayer perceptrons often suffers from plateaus. On a plateau the ODE contribution vanishes. The higher order terms, which contain the dependencies, therefore dominate the learning process. Simulations of a multilayer perceptron with backpropagation learning the tent map demonstrate that dependencies between successive examples can dramatically improve the final learning result. This phenomenon is explained by our analysis, which evidences that randomized learning gets stuck on a plateau, whereas the dependencies in natural learning cause the escape from the plateau. Predictions computed with the theory agree well with the simulations. At the end of

this section we motivated our conjecture that if backpropagation suffers from plateaus, then dependencies (with positive correlations) in example presentation can be helpful, and at least will not do any harm.

At this point, we remark that this paper focuses only on the *learning* process. In practical cases one often has access to a limited number of training data. In such a case, at the global minimum the network model might overfit the data and this training optimum may therefore not be optimal for generalization purposes (Chauvin 1990). Actually, Hochreiter and Schmidhuber (1995) present an algorithm that searches for flat spots to achieve a better generalization. However, issues of generalization and overfitting on a limited set of training examples, though important and interesting, are beyond the scope of the current paper.

For convenience, the paper has been restricted to learning with a constant learning parameter in a stationary environment; i.e., the transition probability $\tau(\boldsymbol{x}|\boldsymbol{x}')$ between successive examples is independent of time. The theory can be extended straightforwardly to learning with time-dependent learning parameters $\eta(t)$ in a changing environment (Benviste *et al.* 1987; Heskes and Kappen 1992), i.e., with a time-dependent transition probability $\tau(\boldsymbol{x}|\boldsymbol{x}';t)$, as long as the time scales of the learning parameter and the changing environment are large compared to the time scale of the learning process, and as long as this last time scale remains large compared to the time scale of the example presentation. As a consequence, time-dependent example selection techniques (Munro 1992; Cachin 1994; Ludik and Cloete 1994), possibly combined with a time-dependent learning parameter, may be devised and evaluated analytically. For instance, one can think of a scheme starting with dependencies designed to avoid plateaus and continuing in a later stage with dependencies designed for the fine tuning around minima. Perhaps such schemes will relate to common sense, like the pedagogical idea that the presentation of examples should start simple and gradually increase in complexity.

In fact, as long as the three previously mentioned time scales remain separated, the theory may also include weight-dependent transition probabilities $\tau(\boldsymbol{x}|\boldsymbol{x}';\boldsymbol{w}.t)$ (Benviste *et al.* 1987). The vector $\boldsymbol{x}$ does not neccessarily represent an example. It may have components describing other fast variables. For instance, fast variables have been utilized to study learning with momentum (Wiegerinck *et al.* 1994), where the adaptation rule does not satisfy equation 2.1. Other obvious candidates for fast variables in neural network theory may be rapidly changing neuron states in recurrent networks. Thus, our framework may be applied to the analysis of the joint dynamics of neurons and weights (Penney *et al.* 1993).

In conclusion, the techniques for the local approximation of stochastic processes with separate time scales prove to be powerful tools for the analysis of on-line learning in neural networks.

Acknowledgments

We thank the referees for their useful suggestions.

References

Amari, S. 1967. A theory of adaptive pattern classifiers. *IEEE Transact. Electronic Comput.* **16**, 299–307.

Barnard, E. 1992. Optimization for training neural nets. *IEEE Transact. Neural Networks* **3** 232–240.

Benviste, A., Metivier, M., and Priouret, P. 1987. *Adaptive Algorithms and Stochastic Approximations.* Springer-Verlag, Berlin.

Brunak, S., Engelbrecht, J., and Knudsen, S. 1990. Cleaning up gene databases. *Nature (London)* **343**, 123.

Cachin, C. 1994. Pedagogical pattern selection strategies. *Neural Networks*, **7**, 175–181.

Chauvin, Y. 1990. Generalization performance of overtrained back-propagation networks. In *Lecture Notes in Computer Science*, L. Almeida and C. Wellekens, eds., Vol 412, pps. 46–55. Springer-Verlag, Berlin.

Finnoff, W. 1994. Diffusion approximations for the constant learning rate backpropagation algorithm and resistance to local minima. *Neural Comp.* **6**, 285–295.

Hansen, L., Pathria, R., and Salamon, P. 1993. Stochastic dynamics of supervised learning. *J. Phys. A* **26**, 63–71.

Haykin, S. 1994. *Neural Networks, A Comprehensive Foundation.* MacMillan, Hamilton, Ontario.

Hertz, J., Krogh, A., and Palmer, R. 1991. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, CA.

Heskes, T. 1994. On Fokker-Planck approximations of on-line learning processes. *J. Phys. A*, **27**, 5145–5160.

Heskes, T., and Kappen, B. 1991. Learning processes in neural networks. *Phys. Rev. A* **44**, 2718–2726.

Heskes, T., and Kappen, B. 1992. Learning-parameter adjustment in neural networks. *Phys. Rev. A* **45**, 8885–8893.

Heskes, T., and Wiegerinck, W. 1996. A theoretical comparison of batch-mode, on-line, cyclic, and almost cyclic learning. *IEEE Trans. Neur. Networks* **7** (4).

Hochreiter, S., and Schmidhuber, J. 1995. Simplifying neural nets by discovering flat minima. In *Advances in Neural Information Processing Systems 7*, G. Tessauro, D. Touretzky, and T. Leen, eds. Morgan Kaufmann, San Mateo, CA.

Hondou, T., and Sawada, Y. 1994. Analysis of learning processes of chaotic time series by neural networks. *Prog. Theoret. Phys.* **91**, 397–402.

Hoptroff, R. 1993. The principles and practice of time series forecasting and business modelling using neural nets. *Neural Comput. Appl.* **1**, 59–66.

Hush, D., Horne, B., and Salas, J. 1992. Error surfaces for multilayer perceptrons. *IEEE Transact. Syst. Man Cybern.* **22**, 1152–1161.

Kohonen, T. 1982. Self-organized formation of topologically correct feature maps. *Biol. Cybern.* **43**, 59–69.

Kuan, C., and White, H. 1994. Artificial neural networks: An econometric perspective. *Economet. Rev.* **13**.

Lapedes, A., and Farber, R. 1988. How neural networks work. In *Neural Information Processing Systems*, D. Anderson, ed., pps. 442–456. American Institute of Physics, New York.

Lee, Y., Oh, S., and Kim, M. 1991. The effect of initial weights on premature saturation in backpropagation learning. *Inte. Joint Conf. Neural Networks*, pps. 765–770. IEEE.

Leen, T., and Moody, J. 1992. Weight space probability densities in stochastic learning: I. Dynamcis and equilibria. In *Advances in Neural Information Processing Systems 5*, S. Hanson, J. Cowan, and L. Giles, eds., pps. 451–458. Morgan Kaufmann, San Mateo, CA.

Ludik, J., and Cloete, I. 1994. Incremental increased complexity training. In *Proceedings of the European Symposium on Artificial Neural Networks '94*, M. Verleysen, ed., pps. 161–165. D Facto, Brussels.

Mpitsos, G., and Burton, M. 1992. Convergence and divergence in neural networks: Processing of chaos and biological analogy. *Neural Networks* **5**, 605–625.

Munro, P. 1992. Repeat until bored: A pattern selection strategy. In *Advances in Neural Information Processing Systems 4*, J. Moody, S. Hanson, and R. Lippman, eds., pps. 1001–1008. Morgan Kaufmann, San Mateo, CA.

Orr, G., and Leen, T. 1992. Weight space probability densities in stochastic learning: II. Transients and basin hopping times. In *Advances in Neural Information Processing Systems 5*, S. Hanson, J. Cowan, and L. Giles, eds., pps. 507–514, Morgan Kaufmann, San Mateo, CA.

Penney, W., Coolen, A., and Sherrington, D. 1993. Coupled dynamics of fast spins and slow interactions in neural networks and spin systems. *J. Phys. A* **26**, 3681–3695.

Radons, G. 1993. On stochastic dynamics of supervised learning. *J. Phys. A* **26**, 3455–3461.

Ritter, H., and Schulten, K. 1988. Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability, and dimension selection. *Biol. Cybern.* **60**, 59–71.

Rumelhart, D., Hinton, G., and Williams, R. 1986. Learning representations by back-propagating errors. *Nature (London)* **323**, 533–536.

Schuster, H. 1989. *Deterministic Chaos*, 2nd rev. ed. VCH, Weinheim.

van Kampen, N. 1992. *Stochastic Processes in Physics and Chemistry*. North-Holland, Amsterdam.

Weigend, A., and Gershenfeld, N., eds. 1993. *Predicting the Future and Understanding the Past: A Comparison of Approaches*. Addison-Wesley, Reading, MA.

Weigend, A., Huberman, B., and Rumelhart, D. 1990. Predicting the future: A connectionist approach. *Int. J. Neural Syst.* **1**, 193–209.

Werbos, P. 1974. Beyond regression: New tools for prediciton and analysis in the behavioral sciences. Ph.D. thesis, Harvard University.

White, H. 1989. Some asymptotic results for learning in single hidden-layer feedforward network models. *J. Amer. Stat. Assoc.* **84,** 1003–1013.

Wiegerinck, W., and Heskes, T. 1994. On-line learning with time-correlated patterns. *Europhy. Lett.* **28,** 451–455.

Wiegerinck, W., Komoda, A., and Heskes, T. 1994. Stochastic dynamics of learning with momentum in neural networks. *J. Phys. A* **27,** 4425–4437.

Wong, F. 1991. Time series forecasting using backpropagation networks. *Neurocomputing* 147–159.

Received January 27, 1995; accepted February 15, 1996.